文章编号:1007-5321(2020)01-0129-06 **DOI**:10.13190/j.jbupt.2019-030

基于卷积神经网络的彩色铅笔画算法

王小玉, 胡鑫豪, 韩昌林
(哈尔滨理工大学 计算机科学与技术学院, 哈尔滨 150080)

摘要: 为了解决传统彩色铅笔画算法生成结果单一的问题,提出了一种基于卷积神经网络(CNN)生成彩色铅笔画的算法. 采用分数阶微分获取原始图像轮廓信息,用卷积神经网络获取艺术家手绘铅笔画风格,利用直方图匹配获取与手绘铅笔画相似的色调,并使用 L-BFGS 优化算法来合成具有铅笔画效果的图像. 该算法能够生成具有不同风格的彩色铅笔画图像. 实验结果表明,该算法生成的图像保留了更多原始图像的细节信息,风格更加灵活多样.

关 键 词: 分数阶微分; 卷积神经网络; 直方图匹配; L-BFGS 优化算法
中图分类号: TP391 **文献标志码**: A

Color Pencil Drawing Based on Convolutional Neural Network

WANG Xiao-yu, HU Xin-hao, HAN Chang-lin
(School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)

Abstract: In order to optimize the single generation result of traditional color pencil drawing algorithms, a convolution neural network (CNN) based color pencil drawing generation method is presented. Fractional differentiation is employed to obtain original image contour information, CNN can obtain pencil drawing style, and histogram matching can obtain similar tones. Meanwhile, L-BFGS algorithm is used to synthesize pencil drawing image. This can generate color pencil drawing images of different styles. Experiments show that the images generated can retain more original image detail information, and feature with more flexible and diverse styles.

Key words: fractional differentiation; convolutional neural network; histogram matching; L-BFGS algorithm

图像非真实感绘制技术[1]不强调图像给人真实的感觉,而是用计算机模拟出艺术风格. 铅笔画绘制是非真实感绘制技术的一种,目前已有很多铅笔画绘制算法. 1999 年,Sousa 等[2]提出一种人机交互的绘制铅笔画算法,但需要大量人力. 2001 年,Mao 等[3]将白噪声图像和向量场作为输入进行线积分卷积运算,得到了比较流畅的图像纹理方向. 2012 年,吴友[4]通过亮度分层和快速线积分卷积方法生成彩色铅笔画,但不能较好地区分图像中亮度差别不大的物体. Lu 等[5]提出了一种模拟艺术家手绘过程的方法,改进了传统的铅笔画算法. 2013 年,谢党恩等[6]用原始图像的灰度值替代单一的黑白像素点,更好地保留了图像的纹理细节,并且抑制了图像颜色失真. 2016 年,孙玉红等[7]通过将图像分割成不同的区域来计算每个区域中铅笔纹理的色调和方向,但过于依赖图像分割方法. 2017 年,潘龙等[8]使用了双色调映射和霓虹变换生成铅笔画图像. 2018 年,王权等[9]提出一种结合 L_0 梯度最小化和 L_1 范数约束的方法展现了图像的层次性,并和原始图像结构保持高度的一致,但是没有考虑到彩色

收稿日期: 2019-02-28
基金项目: 国家自然科学基金项目(60572153, 60972127)
作者简介: 王小玉(1971—), 女, 教授, E-mail:wangxiaoyu@hrbust.edu.cn.

铅笔画具有更加丰富的色调. 传统的铅笔画算法大多基于几何模型和笔画方面,只使用简单的边缘检测算子来对二维图像进行处理[10],由于纹理、噪声等会对图像特征的提取产生影响,所以这些方法并不能得到满意的结果. 另外,传统的铅笔画算法通过一张纹理背景图来模拟铅笔在纸张上反复涂画产生纹理的效果,没有考虑到图像不同区域纹理的区别,并且只能生成固定样式的图像. 现实生活中,艺术家绘制铅笔画的风格是多种多样的,考虑到卷积神经网络(CNN, convolutional neural network)在图像纹理特征提取方面具有良好的效果[11-12],可以模拟出不一致的纹理[13],并且与生物视觉具有一定相似性[14]. 笔者使用分数阶微分算子对原始图像进行更好的边缘检测,提出了利用卷积神经网络提取艺术家手绘铅笔画特征,并使用 L-BFGS 优化算法优化结果得到铅笔画图像的算法,使生成的铅笔画在保留了输入图像原有细节和信息的同时风格也更加的灵活,再通过直方图匹配得到色调图,生成最后的彩色铅笔画.

1 生成轮廓图

生成彩色铅笔画算法的流程如图 1 所示.

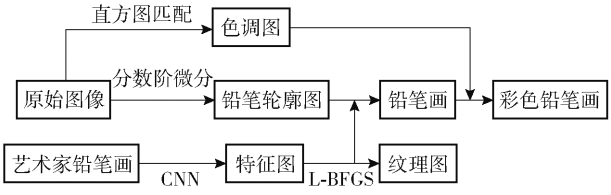


图 1 本文算法流程图

第一步使用分数阶微分算子生成原始图像的轮廓图. 在图像处理的应用当中,相邻像素的灰度值相关性比较高,分数阶微分能够非线性地加强具有复杂细节信息的图像高频部分,并且增强灰度级变化不明显的区域中的纹理细节,同时也能够较好保留图像平滑区域中的低频轮廓特征[15]. 由 G-L 定义的分数阶微分的形式为

$$ {}_a^G D_t^v = \lim_{h\to 0}\frac{1}{h^v}\sum_{m=0}^{\frac{t-a}{h}}(-1)^m\frac{\Gamma(v+1)}{m!\ (v-m+1)}s(t-mh) \tag{1} $$

其中:G 表示 G-L 定义的分数阶微分,v 表示第 v 阶的导数,a 和 t 表示持续时间$[a,t]$,h 表示$[a,t]$之间的等分间隔,$(t-a)/h$ 表示等分之后一共有多少个区间,m 表示第几个等分区间,Γ 表示伽马函数.

对于二维图像信号 $s(x,y)$,有以下 2 个表达式.

$$ \begin{aligned} \frac{\partial^v s(x,y)}{\partial x^v} \cong & s(x,y)+(-v)s(x-1,y)+\\ & \frac{(-v)(-v+1)}{2}s(x-2,y)+\\ & \frac{(-v)(-v+1)(-v+2)}{6}s(x-3,y)+\cdots+\\ & \frac{\Gamma(-v+1)}{m!\ \Gamma(-v+m+1)}s(x-m,y) \end{aligned} \tag{2} $$

$$ \begin{aligned} \frac{\partial^v s(x,y)}{\partial y^v} \cong & s(x,y)+(-v)s(x,y-1)+\\ & \frac{(-v)(-v+1)}{2}s(x,y-2)+\\ & \frac{(-v)(-v+1)(-v+2)}{6}s(x,y-3)+\cdots+\\ & \frac{\Gamma(-v+1)}{m!\ \Gamma(-v+m+1)}s(x,y-m) \end{aligned} \tag{3} $$

由式(2)和式(3)可以得出前 m 项的系数:

$$ \left.\begin{aligned} a_0 &= 1\\ a_1 &= -V\\ a_2 &= \frac{(-v)(-v+1)}{2!}\\ &\vdots\\ a_m &= \frac{\Gamma(-v+1)}{m!\ \Gamma(-v+m+1)} \end{aligned}\right\} \tag{4} $$

为了实现运算误差不大并且运算过程不太复杂的分数阶滤波器,取差分表达式的前 3 项系数构造 5×5 大小的分数阶掩膜,并且进行归一化处理. 为了获得良好的抗旋转能力,选择 8 个对称方向:正负 x 坐标,正负 y 坐标,左上对角线,左下对角线,右上对角线,右下对角线. 8 个方向上的分数阶归一化掩膜算子如图 2 所示.

$$ K = 8a_0 + (4+2\sqrt{2})a_1 + (4+2\sqrt{2})a_2 \tag{5} $$

$\frac{a_2}{\sqrt{2}k}$		$\frac{a_2}{k}$		$\frac{a_2}{\sqrt{2}k}$
	$\frac{a_1}{\sqrt{2}k}$	$\frac{a_1}{k}$	$\frac{a_1}{\sqrt{2}k}$	
$\frac{a_2}{k}$	$\frac{a_1}{k}$	$\frac{8a_0}{k}$	$\frac{a_1}{k}$	$\frac{a_2}{k}$
	$\frac{a_1}{\sqrt{2}k}$	$\frac{a_1}{k}$	$\frac{a_1}{\sqrt{2}k}$	
$\frac{a_2}{\sqrt{2}k}$		$\frac{a_2}{k}$		$\frac{a_2}{\sqrt{2}k}$

图 2 5×5 分数阶微分掩膜算子

直接获得的边缘检测图像并不能用来做线条轮廓图,还需考虑到手绘铅笔画的方向. 接下来使用

卷积来塑造线条的方向感. 选择边缘检测图中正负 x 坐标,正负 y 坐标,左上对角线,左下对角线,右上对角线,右下对角线 8 个方向进行卷积,公式为

$$Y_i = L_i Y \tag{6}$$

其中:L_i 的长度为图像的高度或者宽度的 1/30,选择像素中卷积结果最大的方向最为响应 C_i,其余的方向置 0. 然后再通过一次卷积来塑造出线条轮廓图,公式为

$$S = \sum_{i=1}^{8} (L_i \otimes C_i) \tag{7}$$

2　获取铅笔画纹理

2.1　CNN 提取铅笔画特征

在传统的铅笔画纹理模拟中,Lu 等[5]采用的方法是用一张背景纹理图,并使用指数组合的方式和共轭梯度法来求解反复涂画的最优次数,模拟出艺术家真实手绘铅笔画的纹理效果. 考虑到 CNN 在每一个卷积层都包含若干个特征图,笔者利用 CNN 处理输入图像时产生的特征图来模拟纹理效果,并且使用 L-BFGS 优化算法求出最优解. 本文算法在 CNN 中只使用卷积层和池化层,不使用到全连接层. 卷积层中的卷积滤波器大小为 $3 \times 3 \times k$,k 为特征图的数目,池化层采用最大化池,大小为 2×2.

使用一张艺术家手绘的铅笔画图像作为 CNN 的输入,利用 Gram 矩阵来计算损失函数. Gram 矩阵是每层特征图之间的内积,在特征图中,每个数字都来自于一个特定滤波器在特定位置的卷积,因此每个数字代表一个特征的强度,而 Gram 矩阵计算的是特征之间的相关性. 同时,Gram 矩阵中的对角线元素体现了每个特征在图像中出现的量,因此,Gram 矩阵可以代表图像的大体风格. 假设每个卷积层 l 中有 N_l 个过滤器,则每一个卷积层中有 N_l 个特征图,在第 l 层的 j 处,第 i 个滤波器的激活函数为 f_{ij}^l,则在第 l 层中的响应为 F^l,$F^l \in R^{N_l \times M_l}$,$R^{N_l \times M_l}$ 为全体 $\boldsymbol{M} \times \boldsymbol{N}$ 矩阵所构成的空间. 基于 Gram 矩阵的特征图空间相关性公式为

$$G_{ij}^{sl} = \sum_k F_{ki}^l F_{kj}^l \tag{8}$$

其中:s 表示特征图空间相关性,k 表示通过第 k 个滤波器的特征图. 基于不同滤波器响应之间的相关性公式为

$$G_{ij}^{fl} = \sum_k F_{ik}^l F_{jk}^l \tag{9}$$

由式(8)和式(9)可以看出,$\boldsymbol{G}^{sl} = (\boldsymbol{G}^{fl})^{\mathrm{T}}$. 设输入的艺术家手绘铅笔画为 s,需要生成的铅笔画图像为 t,则在 l 层的损失函数为

$$E_l = \frac{1}{8N_l^2 M_l^2} \sum_{i,j} ((S_{ij}^{sl} - T_{ij}^{sl})^2 + (S_{ij}^{fl} - T_{ij}^{fl})^2) \tag{10}$$

其中:S_{ij}^{sl}、T_{ij}^{sl}、S_{ij}^{fl}、T_{ij}^{fl} 为 s 和 t Gram 矩阵中的元素,且 $\boldsymbol{S}^{sl} = (\boldsymbol{S}^{fl})^{\mathrm{T}}$,$\boldsymbol{T}^{sl} = (\boldsymbol{T}^{fl})^{\mathrm{T}}$. 化简式(10)得

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (S_{ij}^l - T_{ij}^l)^2 \tag{11}$$

基于 Gram 矩阵的总损失为

$$E(s,t) = \sum_l w_l E_l \tag{12}$$

其中 w_l 为每一层损失对总损失的权值.

2.2　生成铅笔画纹理图像

在 2.1 节已经得到了基于 Gram 矩阵的总损失函数公式,为了得到更好的结果,采用 L-BFGS 优化算法优化 $f(t)$,设

$$f(t) = E(s,t) = \sum_l w_l E_l \tag{13}$$

求出使 $f(t)$ 最小化的 t,即是生成的铅笔画风格图像. L-BFGS 优化算法是一种最优化方法[16],与牛顿算法不同的是,牛顿算法需要计算二阶倒数构成的 Hesse 矩阵,因此需要大量的存贮空间和计算量. L-BFGS 优化算法只需要构造 Hesse 矩阵的近似矩阵,因此计算量比牛顿算法大大减小,同时只保留和利用最近的 m 次迭代信息,在存储空间上也比牛顿算法减少了很多. 对要优化的函数 $f(x)$ 在 x_{i+1} 附近的二阶泰勒展开式两边求导为

$$g(x) \approx g(x_{i+1}) + \boldsymbol{G}(x_{i+1})(x - x_{i+1}) \tag{14}$$

其中:$g(x_{i+1})$ 为函数在 x_{i+1} 处的梯度,$\boldsymbol{G}(\boldsymbol{x}_{i+1})$ 为函数在 x_{i+1} 处的 Hesse 矩阵. 若 $\boldsymbol{H}(\boldsymbol{x}_{i+1})$ 为 $\boldsymbol{G}(\boldsymbol{x}_{i+1})$ 的逆矩阵,则

$$\boldsymbol{H}(x_{i+1}) y_i \approx z_i \tag{15}$$

其中:$y_i = g(x_{i+1}) - g(x_i)$,$z_i = x_{i+1} - x_i$.

在 L-BFGS 优化算法中基于 Hesse 逆矩阵的递推公式为

$$\boldsymbol{H}_{i+1} = \boldsymbol{V}_i^{\mathrm{T}} \boldsymbol{H}_i \boldsymbol{V}_i + p_i z_i \boldsymbol{z}_i^{\mathrm{T}} \tag{16}$$

其中:$p_i = \dfrac{1}{\boldsymbol{y}_i^{\mathrm{T}} z_i}$,$\boldsymbol{V}_i = (\boldsymbol{I} - p_i z_i \boldsymbol{y}_i^{\mathrm{T}})$,初始值 $\boldsymbol{H}_i^0 = \dfrac{\boldsymbol{z}_i^{\mathrm{T}} y_i}{\| y_i \|^2} \boldsymbol{I}$,$\boldsymbol{I}$ 为单位矩阵. 在迭代 m 次后的 $\boldsymbol{H}_i$ 为

$$\boldsymbol{H}_i = \boldsymbol{V}_{i-1}^{\mathrm{T}} \boldsymbol{H}_i \boldsymbol{V}_{i-1} + p_{i-1} z_{i-1} \boldsymbol{z}_{i-1}^{\mathrm{T}} =$$

$$\vdots$$

$$(\boldsymbol{V}_{i-1}^{\mathrm{T}} \cdots \boldsymbol{V}_{i-m}^{\mathrm{T}}) \boldsymbol{H}_i^0 (\boldsymbol{V}_{i-m} \cdots \boldsymbol{V}_{i-1}) +$$

$$p_{i-m} (\boldsymbol{V}_{i-1}^{\mathrm{T}} \cdots \boldsymbol{V}_{i-m+1}^{\mathrm{T}}) z_{i-m} \boldsymbol{z}_{i-m}^{\mathrm{T}} (\boldsymbol{V}_{i-m-1} \cdots \boldsymbol{V}_{i-1}) + \cdots +$$

$$p_{i-1}z_{i-1}z_{i-1}^{\mathrm{T}} \tag{17}$$

对式(13)中的$f(t)$求梯度：

$$\frac{\partial f}{\partial t}=\frac{\partial E(s,t)}{\partial t}=\frac{\partial E(s,t)}{\partial S_{ij}^{l}}=\sum_{l} w_l \frac{\partial E_l}{\partial S_{ij}^{l}}=$$

$$\begin{cases}\sum_{l} w_l \frac{1}{N_l^2 M_l^2}\left((\boldsymbol{S}^{l})^{\mathrm{T}}(\boldsymbol{S}^{l}-\boldsymbol{T}^{l})\right)_{ij}, & \text{if } S_{ij}^{l}>0\\ 0, & \text{if } S_{ij}^{l}\leqslant 0\end{cases} \tag{18}$$

将$f(t)$和$\frac{\partial f}{\partial t}$的10次迭代值发送到L-BFGS模型中，此时$y_i=\frac{\partial f}{\partial t_{i+1}}-\frac{\partial f}{\partial t_i}$，$z_i=f(t_{i+1})-f(t_i)$，即可计算出满足$f(t)$最小化的值$t$. 用随机初始化后的$f(t_1)$，$\frac{\partial f_1}{\partial t_1}$可以计算出式(16)中的$\boldsymbol{H}_1$，由式(17)可以计算出$t_2$，设置最大迭代次数为10 000，迭代进行10 000次或$\boldsymbol{H}_i$的值几乎不发生变化，即可生成艺术家手绘铅笔画的风格图像. 最后将风格图像和边缘图像结合，生成铅笔画图像.

3 生成彩色铅笔画图像

想要得到彩色铅笔画，还需要生成原始图像的色调图. 颜色直方图对图像处理有着重要意义，描述了图像中颜色的数量特征，反映出颜色的统计分布和基本色调. 不同的图像可能有相同的颜色分布，因此具有相同的直方图. 艺术家手绘彩色铅笔画的直方图与自然图像的直方图存在一定差别. 在手绘铅笔画中，图像大致可以分成3个区域，一个是用笔画反复勾勒形成的黑暗层，第2个是偏白色的明亮层，第3个黑暗层向明亮层过度的中间层. Lu等[5]使用一种有参数的模型来表示手绘铅笔画的色调分布，公式为

$$p(r)=\frac{1}{Z}\sum_{i=1}^{3}\omega_i p_i(r) \tag{19}$$

其中：r表示色调值；$p(r)$表示图像中像素值是r的概率；Z表示归一化因子，使$p(v)$从0～1的积分为1；i表示3个色调层；ω_i表示在每个色调层中像素数量的权重. 经过对铅笔画直方图的研究观察，明亮层大致服从拉普拉斯分布，中间层大致服从均匀分布，黑暗层大致服从高斯分布. 因此，将3个直方图通过不同的权重映射到一个设定的直方图中，得到较好的色调图. 使用下面3个公式描述3种分布：

$$P_1(r)=\begin{cases}\frac{1}{\sigma_b}\mathrm{e}^{\frac{1-r}{\sigma_b}}, & r\leqslant 1\\ 0, & r>1\end{cases} \tag{20}$$

$$P_2(r)=\begin{cases}\frac{1}{u_a-u_b}, & u_a\leqslant r\leqslant u_b\\ 0, & r<u_a \text{ 或 } r>u_b\end{cases} \tag{21}$$

$$P_3(r)=\frac{1}{\sqrt{2\pi}\sigma_d}\mathrm{e}^{-\frac{(r-\mu_d)^2}{2\sigma_d^2}} \tag{22}$$

式(20)表示拉布拉斯分布，σ_b代表分布尺度. 式(21)表示均匀分布，u_a，u_b用来控制分布范围. 式(22)表示高斯分布，μ_d表示阴暗笔画平均值，σ_d是一个比例参数. 3个公式中，各个参数确定了色调直方图的形状. 首先在明亮层和黑暗层中，根据阈值来手动划分，其余像素在中间层中. 然后可以采用最大似然估计来估计各个参数值. 设像素平均值和标准差为p和q，则计算参数的公式如下：

$$\sigma_b=\frac{1}{N}\sum_{i=1}^{N}|x_i-1| \tag{23}$$

$$u_a=p_m-\sqrt{3}q_m \tag{24}$$

$$u_b=p_m+\sqrt{3}q_m \tag{25}$$

$$\mu_d=p_d \tag{26}$$

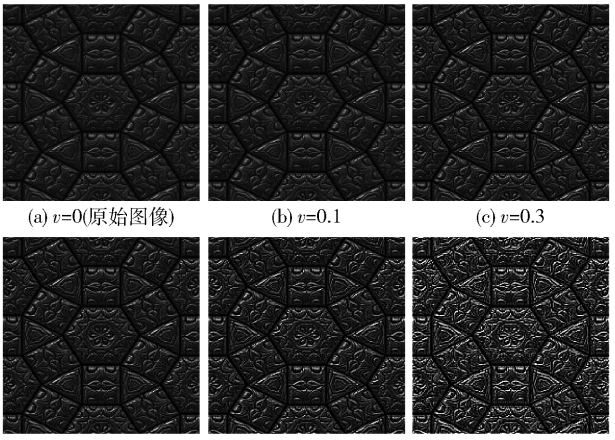
$$\sigma_d=q_d \tag{27}$$

其中：x表示像素值，N表示像素数量. 用最大似然估计方法可以计算出参数值. 其中σ_b值为9，u_a值为105，u_b值为225，μ_d值为90，σ_d值为11. 经实验后得出，明亮层的权重参数设置为50，中间层的参数设置为40，阴暗层的参数设置为10，然后用直方图匹配的方法生成色调图，最后与上一步的铅笔画图像结合得到最后的彩色铅笔画效果.

4 实验结果与分析

实验在一台i5处理器，4 GB内存，AMD radeon HD 8600M显卡，64位win7操作系统的PC机上，在tensorflow环境中使用python编程实现. 首先选择了一幅纹理细节较多的图片进行$0\leqslant v\leqslant 1$范围内的多尺度分数阶微分，目的是从多尺度的角度来分析当v的值为多少时，可以使图像具有更好的边缘检测效果，突出更多的纹理细节. 实验结果如图3所示.

图3显示了不同尺度的分数阶微分对纹理增强的能力. 从图中可以看出，当$v=0$时，图像并没有产生变化. 当$v=1$时，实际上是一阶微分，图像的高频边缘部分出现了大量白色，无法分辨具体的细节. 随着v的增大，分数阶微分的效果越来越接近一阶微分，当$v=0.9$时，高频边缘的白色已经开始出现，所以分数阶微分取$v=0.7$时效果最好.



(a) v=0(原始图像)　(b) v=0.1　(c) v=0.3

(d) v=0.5　(e) v=0.7　(f) v=0.9

图 3　图像的不同阶分数阶微分

在生成色调图中需要设置 3 个色调层的权重，设明亮层的权重为 ω_1，中间层的权重为 ω_2，阴暗层的权重为 ω_3. 实验中选取一张 960 × 600 的图像，分别设置了 3 组不同的权重，实验结果如图 4 所示. 从图 4 中可以看出，明亮层的权重越大，则色调图的亮度越大，阴暗层的权重越大，则色调图越深. 当 $\omega_1 = 50, \omega_2 = 40, \omega_3 = 10$ 时，色调图的效果最好.

(a) 权重比40:30:30　(b) 权重比50:40:10　(c) 权重比70:28:2

图 4　不同权重对实验结果的影响

在使用 CNN 提取纹理特征生成铅笔画的部分，选择了一张人物照片和一张建筑照片进行实验. 实验结果如图 5 所示，图(a)是原始人物图像，(b)和(d)是输入的人物样式图像，(c)和(e)是生成的人物铅笔画结果，(f)是原始的建筑图像，(g)和(i)是输入的建筑样式图像，(h)和(j)是生成的建筑铅笔画结果. 可以看出，本文算法可以根据输入样式图像的不同，生成具有相似样式的铅笔画结果，并能保留原始图像的轮廓和细节信息. 而传统的铅笔画绘制算法只能生成固定样式的铅笔画，所以本文算法生成铅笔画的算法比较灵活.

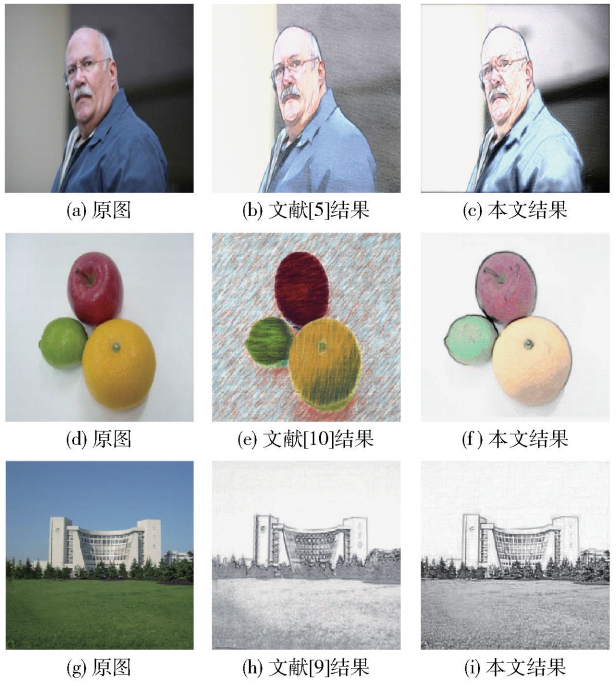
图 6 所示为本文结果和其他结果的比较. 可以看出，在人物彩色铅笔效果中，Lu 等[5]的方法虽然有不错的效果，但是丢失了一些衣服上的细节，本文算法的结果中，人物衣服上具有更加丰富的细节纹理. 在水果图中，Yamamoto 等[10]的方法虽然有彩色

(a) 人物图　(b) 人物样式1　(c) 结果1　(d) 人物样式2

(e) 结果2　(f) 建筑图　(g) 建筑样式1　(h) 结果1

(i) 建筑样式2　(j) 结果2

图 5　本文算法结果



(a) 原图　(b) 文献[5]结果　(c) 本文结果

(d) 原图　(e) 文献[10]结果　(f) 本文结果

(g) 原图　(h) 文献[9]结果　(i) 本文结果

图 6　实验结果与其他算法对比

素描的效果，但是水果上面的纹理细节比较模糊，没有很好地展现出来，从本文的实验结果能够很明显地看出苹果把等纹理细节，具有更好的手绘素描效果. 在结构复杂的建筑图中，用王权等[9]的方法虽然有铅笔画效果，但建筑物窗户的线条模糊，树木也缺少细节，本文算法的结果中窗户线条轮廓更加清晰，树木也有较好的纹理，具有更好的效果.

5 结束语

参考了传统彩色铅笔画生成方法,发现基于笔画和纹理模拟的方法较多,生成的彩色铅笔画结果只有一种. 考虑到卷积神经网络对纹理特征的提取具有很好的效果,利用分数阶微分提取原始图像的边缘,通过卷积神经网络特征响应来模拟纹理,并结合色调图实现彩色铅笔画风格的实现. 实验结果表明,该算法产生的纹理效果逼真柔和,能够保留更多的细节信息,同时能够生成不同样式的结果,是一种有效的非真实感绘制方法.

本文算法虽然能产生较好的彩色铅笔画效果,但依然存在不足之处,也是以后的学习研究中需要进一步优化的方向,例如 L-BFGS 优化算法的效率还需要优化. 当输入的样式图像和原始图像差别很大时,生成结果需要的时间更长,并且会产生伪影,这也是以后需要优化的研究方向.

参考文献:

[1] Strassmann S. Hairy brushes[J]. Computer Graphics, 1986, 20(4): 225-232.

[2] Sousa M C, Buchanan J W. Observational model of blenders and erasers in computer-generated pencil rendering[C]// Conference on Graphics Interface. Kingston: Morgan Kaufmann Publishers Inc , 1999: 157-166.

[3] Mao Xiaoyang, Nagasaka Y, Imamiya A. Automatic generation of pencil drawing from 2D images using line integral convolution[C]// Proceedings of the 7th International Conference on Computer Aided Design and Computer Graphics. New York: IEEE Press, 2001: 240-248.

[4] 吴友. 基于图像的彩色铅笔画快速生成算法研究[D]. 长沙: 长沙理工大学, 2012.

[5] Lu Cewu, Xu Li, Jia Jiaya. Combining sketch and tone for pencil drawing production[C]// Proceedings of the Symposium on Non-Photorealistic Animation and Rendering. New York: ACM, 2012: 65-73.

[6] 谢党恩, 张志立, 徐丹. 一种改进的二维彩色铅笔画自动绘制算法[J]. 计算机应用与软件, 2013, 30(8): 28-31.
Xie Dangen, Zhang Zhili, Xu Dan. An improved automatic rendering algorithm for two-dimensional color pencil drawings [J]. Computer Applications and Software, 2013, 30(8): 28-31.

[7] 孙玉红, 张元科, 孟静. 基于纹理和草图的图像铅笔画绘制[J]. 计算机应用, 2016, 36(7): 1976-1980.
Sun Yuhong, Zhang Yuanke, Meng Jing. Image pencil drawing based on texture and sketch [J]. Computer Applications, 2016, 36(7): 1976-1980.

[8] 潘龙, 纪庆革, 陈靖. 线积分卷积与双色调映射相结合的彩色素描模拟方法[J]. 中国图像图形学报, 2017, 22(7): 875-885.
Pan Long, Ji Qingge, Chen Jing. A color sketch simulation method combining line integral convolution with two-tone mapping [J]. Chinese Journal of Image and Graphics, 2017, 22(7): 875-885.

[9] 王权, 胡越黎, 燕明, 等. 基于混合梯度最小化的铅笔画生成[J]. 上海大学学报(自然科学版), 2018, 24(2): 168-175.
Wang Quan, Hu Yueli, Yan Ming, et al. Pencil painting generation based on mixed gradient minimization [J]. Journal of Shanghai University (Natural Science Edition), 2018, 24(2): 168-175.

[10] Yamamoto S, Mao Xiaoyang, Imamiya A. Enhanced LIC pencil filter[C]// International Conference on Computer Graphics, Imaging and Visualization. New York: IEEE Press, 2004: 251-256.

[11] 冀中, 刘青, 聂林红, 等. 基于卷积神经网络的纹理分类方法研究[J]. 计算机科学与探索, 2016, 10(3): 389-397.
Ji Zhong, Liu Qing, Nie Linhong, et al. Texture classification method based on convolution neural network [J]. Computer Science and Exploration, 2016, 10(3): 389-397.

[12] Xiao Tianjun, Zhang Jiaxing, Yang Kuiyuan, et al. Error-driven incremental learning in deep convolutional neural network for large-scale image classification[C]// Proceedings of the 22nd ACM International Conference on Multimedia. New York: ACM, 2014: 177-186.

[13] Cai Xiuxia, Song Bin. Combining inconsistent textures using convolutional neural networks[J]. Journal of Visual Communication and Image Representation, 2016, 40: 366-375.

[14] Yamins D, Hong H, Cadieu C, et al. Performance-optimized hierarchical models predict neural responses in higher visual cortex[J]. Proceedings of the National Academy of Sciences, 2014, 111(23): 8619-8624.

[15] 李军成. 图像边缘检测的分数阶微分算子研究[J]. 计算机应用与软件, 2015, 32(12): 206-209.
Li Juncheng. Fractional differential operators for image edge detection [J]. Computer Applications and Software, 2015, 32(12): 206-209.

[16] Zhu Ciyou, Byrd R H, Lu Peihuang, et al. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization[J]. ACM Transactions on Mathematical Software, 1997, 23(4): 550-560.